Docket No.: 56937-111 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	Customer Number: 20277
Katsuya SHINOHARA	:	Confirmation Number:
Serial No.:	:	Group Art Unit:
Filed: April 01, 2004	:	Examiner: Unknown

For: SIMULATOR APPARATUS AND RELATED TECHNOLOGY

CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Mail Stop CPD
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicant hereby claims the priority of:

Japanese Patent Application No. 2003-104907, filed April 9, 2003

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Michael E. Fogarty
Registration No. 36,139

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 MEF:tlb
Facsimile: (202) 756-8087
Date: April 1, 2004

56937-111
SHINOHARA
April 1, 2004

日本国特許庁
JAPAN PATENT OFFICE

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　４月　９日

出願番号
Application Number: 特願２００３－１０４９０７

[ST. 10/C]: [ＪＰ２００３－１０４９０７]

出願人
Applicant(s): 松下電器産業株式会社

２００３年１１月２１日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３０９６６５８

【書類名】　特許願
【整理番号】　5037540160
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 17/50
F24D 13/02
【発明者】
【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
【氏名】　篠原　克哉
【特許出願人】
【識別番号】　000005821
【氏名又は名称】　松下電器産業株式会社
【代理人】
【識別番号】　100086737
【弁理士】
【氏名又は名称】　岡田　和秀
【電話番号】　06-6376-0857
【手数料の表示】
【予納台帳番号】　007401
【納付金額】　21,000円
【提出物件の目録】
【物件名】　明細書　1
【物件名】　図面　1
【物件名】　要約書　1
【包括委任状番号】　9305280
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　シミュレータ装置、シミュレーション方法および性能解析方法

【特許請求の範囲】

【請求項１】　被シミュレーション対象システムを構成するＣＰＵ、前記ＣＰＵや周辺のハードウェアのブロックを接続するバスおよび前記ハードウェアをシミュレートするシミュレータ装置であって、前記ＣＰＵのモデル、バス、および前記バスに接続される前記ハードウェアのモデルとのインターフェースを実装し、前記インターフェースを用いて、前記被シミュレーション対象システムの性能／機能検証を実行することを特徴とするシミュレータ装置。

【請求項２】　前記各モデルのインターフェースが、ソフトウェア開発検証用システムシミュレータで使用できるインターフェースを備えていることを特徴とする請求項１に記載のシミュレータ装置。

【請求項３】　前記各モデルのインターフェースが、ハードウェア検証用システムシミュレータで使用できるインターフェースを備えていることを特徴とする請求項１に記載のシミュレータ装置。

【請求項４】　前記各モデルのインターフェースが、システム検証用システムシミュレータで使用できるインターフェースを備えていることを特徴とする請求項１に記載のシミュレータ装置。

【請求項５】　前記各モデルのインターフェースが、デバッグで使用できるインターフェースを備えていることを特徴とする請求項１に記載のシミュレータ装置。

【請求項６】　請求項５に記載のインターフェースを用いて、デバッグすることを特徴とするシミュレーション方法。

【請求項７】　システムの１クロックサイクル数分のシミュレーションを実行するためのインターフェースを備えていることを特徴とする請求項１に記載のシミュレータ装置。

【請求項８】　請求項７に記載の前記シミュレータ装置を用いてシステムの１クロックサイクル数分のシミュレーションを実行することを特徴とするシミュレーション方法。

【請求項９】　システムのシミュレーション時間分のシミュレーションを実行するためのインターフェースを備えていることを特徴とする請求項１に記載のシミュレータ装置。

【請求項１０】　請求項９に記載の前記インターフェースを用いて、システムのシミュレーション時間分のシミュレーションを実行することを特徴とするシミュレーション方法。

【請求項１１】　前記各モデルのインターフェースが、性能解析で使用できる拡張用インターフェースを備えていることを特徴とする請求項１に記載のシミュレータ装置。

【請求項１２】　請求項１１に記載の前記ンターフェースを用いて、性能解析することを特徴とするシミュレーション方法。

【請求項１３】　請求項１１に記載の前記インターフェースを用いて生成された情報を基に、性能解析することを特徴とする性能解析方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、大規模集積回路（ＬＳＩ）設計、特にシステムオンチップ化に対応するための設計に関するものであり、設計上流におけるハードウェア／ソフトウェア／システムの検証、開発、評価を行うためのシミュレータ装置、シミュレーション方法および性能解析方法に関するものである。

【０００２】

【従来の技術】

従来から、電子機器は、例えばプロセッサ、メモリ、バス等の種類ごとのＬＳＩを、個別に半導体チップ上に形成した後、各チップをプリント配線基板などの母基板上に実装することによって製造されてきた。

【０００３】

ところが、近年、これに用いられる集積回路装置に対し、１チップ化、小型化、軽量化、省電力化および低コスト化の要求が高まっている。このような傾向は、特にデジタル情報家電分野において、より顕著にみられる。そして、これに応

じて、半導体メーカーはその研究開発の重心をメモリからシステムＬＳＩへと移行してきている。

【０００４】

システムはソフトウェアとハードウェアから構成される。ハードウェアとソフトウェアに分けられた後、ハードウェアとソフトウェアを逐次的に開発していた。このようにハードウェアとソフトウェアを逐次的に開発すると、開発期間の増大につながるため、ハードウェアとソフトウェアを同時に並行開発するハードウェア／ソフトウェア協調設計という手法がとられつつある。ハードウェアの検証用にシステムシミュレータ、エミュレータを用い、ソフトウェアの開発／検証用にも別途、システムシミュレータや評価ボードを用いて、それぞれの開発／検証を行うのである。

【０００５】

従来のシミュレーション技術は、異なる用途向けの、異なる抽象度で記述されたモデルを想定している（例えば、特許文献１参照）。

【０００６】

また、互いに異なる抽象度で書かれたモデルを統一するシミュレーション方法も提案されている（例えば、特許文献２参照）。

【０００７】

【特許文献１】

特開２００２－１５７２９１号公報（第６－７頁、第２図）（半導体集積回路装置の設計方法）

【特許文献２】

特願２００２－１６２０４８号（シミュレータ装置並びにシミュレーションモデル生成プログラム）

【０００８】

【発明が解決しようとする課題】

しかしながら、ハードウェア／ソフトウェア協調設計を行うためには、そのための設計環境が必要となる。すなわち、システムシミュレータが必要となる。ハードウェアとソフトウェアの分割に使用されるシステム検証用シミュレータと、

ハードウェアの検証に使用されるハードウェア検証用システムシミュレータと、ソフトウェアの開発検証用に用いられるソフトウェア開発検証用シミュレータと、それぞれの用途向けに、プロセッサ、バス、ハードウェアモデルを用意するのが現状である。

【０００９】

しかし、システムＬＳＩの複雑化・大規模化を鑑みると、それらのモデルを用意するための工数も同様に増加の一途をたどっている。結果、ハードウェア／ソフトウェア協調設計を行うことで得られる恩恵と同程度かそれ以上の工数が発生し、ハードウェア／ソフトウェア協調設計の利点が失われてしまう。

【００１０】

本発明は上記問題点に鑑み、それぞれのシステムシミュレータで利用できる共通のモデルを用いたシミュレータ装置、シミュレーション方法、性能解析方法を提供することを目的とする。

【００１１】

【課題を解決するための手段】

以上の目的を達成するための本発明のシステムシミュレータ装置およびシミュレーション方法は、大規模なシステムＬＳＩの１チップシミュレーションで用いるブロックについて、異なるシミュレータの制御部からブロックを呼び出し可能なインターフェースを備えたものである。また、デバッグ用のインターフェースを実装することで、各用途のための検証解析機能を実現するものである。以下、順次説明する。

【００１２】

本発明によるシミュレータ装置は、被シミュレーション対象システムを構成するＣＰＵ、前記ＣＰＵや周辺のハードウェアのブロックを接続するバスおよび前記ハードウェアをシミュレートするシミュレータ装置であって、前記ＣＰＵのモデル、バス、および前記バスに接続される前記ハードウェアのモデルとのインターフェースを実装し、前記インターフェースを用いて、前記被シミュレーション対象システムの性能／機能検証を実行することを特徴とする。

【００１３】

上記構成において、前記各モデルのインターフェースとしては、ソフトウェア開発検証用システムシミュレータで使用できるインターフェースや、ハードウェア検証用システムシミュレータで使用できるインターフェースや、システム検証用システムシミュレータで使用できるインターフェースなどがある。

【００１４】

この構成によれば、ＣＰＵのモデルとハードウェアのモデルを構成するとともに、両モデル間にインタフェースを介在させていて、ソフトウェア検証向け、システム検証向け、ハードウェア検証向けなど各種の検証向けにモデルを共通化することができる。すなわち、大規模なシステムＬＳＩのハードウェア検証用システムシミュレータ、ソフトウェア開発検証用シミュレータ、システム検証用システムシミュレータをはじめとする各種目的用システムシミュレータにおいて、共通のモデルが利用可能になるため、異なるシミュレータ間でモデルの共有化が図られ、統合シミュレーション制御方法によって、適切な精度のシミュレーションを同一のモデルを用いて実行できる。したがって、それぞれの目的ごとにモデルを用意する必要がなくなる。

【００１５】

また、上記シミュレータ装置の構成において、前記各モデルのインターフェースとしては、デバッグで使用できるインターフェースもある。これに対応するシミュレーション方法は、そのデバッグ用のインターフェースを用いてデバッグすることである。例えば、ブロックの機能としてデータ転送の開始、終了といった処理が行われた場合にブレークを行うようにするデバッグ用のインターフェースを備えることで、各用途向けモデルで共通して利用でき、デバッグ性が向上する。

【００１６】

さらに、上記シミュレータ装置の構成において、前記各モデルのインターフェースとしては、システムの１クロックサイクル数分のシミュレーションを実行するためのインターフェースもある。これに対応するシミュレーション方法は、そのシステムの１クロックサイクル数分のシミュレーションを実行するためのインターフェースを用いて、システムの１クロックサイクル数分のシミュレーション

を実行することである。各モデルを駆動するタイミングがクロックレベルであるときに有効である。

【００１７】

また、上記シミュレータ装置の構成において、前記各モデルのインターフェースとしては、システムのシミュレーション時間分のシミュレーションを実行するためのインターフェースもある。これに対応するシミュレーション方法は、そのシステムのシミュレーション時間分のシミュレーションを実行するためのインターフェースを用いて、システムのシミュレーション時間分のシミュレーションを実行することである。各モデルで使用するクロックサイクルが異なる場合に有効である。すなわち、異なるクロックソースを用いるモデル混在のシミュレーション環境が実現できる。

【００１８】

また、上記シミュレータ装置の構成において、前記各モデルのインターフェースとしては、性能解析で使用できる拡張用インターフェースもある。これに対応するシミュレーション方法または性能解析方法は、その解析で使用できる拡張用インターフェースを用いて解析することである。性能解析の結果を基にシステムをチューニングして、最適なシステムを設計することが可能となる。

【００１９】

【発明の実施の形態】

以下、本発明の一実施形態について、図面を参照しながら説明する。

【００２０】

図１は本発明のシミュレータ装置の構成を示すブロック図である。図中、検証用シミュレータ装置１は、被シミュレーション対象システムのシミュレータ装置である。被シミュレーション対象システムは、ブロック３１、ブロック３２、ブロック３３から構成された場合の例である。ブロック３１のモデルは、ソフトウェア検証シミュレータ用Ｉ／Ｆ３と、ハードウェア検証用シミュレータ用Ｉ／Ｆ４と、システム検証シミュレータ用Ｉ／Ｆ５と、デバッグ用Ｉ／Ｆ６と、拡張用Ｉ／Ｆ７とを備え、ブロックの機能モデル８からなるシミュレータモデル９として、検証用シミュレータ装置１を構成する。

【００２１】

同様に、ブロック３２のモデルは、ソフトウェア検証シミュレータ用Ｉ／Ｆ１０と、ハードウェア検証用シミュレータ用Ｉ／Ｆ１１と、システム検証シミュレータ用Ｉ／Ｆ１２と、デバッグ用Ｉ／Ｆ１３と、拡張用Ｉ／Ｆ１４とを備え、ブロックの機能モデル１５からなるシミュレータモデル１６として、検証用シミュレータ装置１を構成する。

【００２２】

さらに、ブロック３３のモデルは、ソフトウェア検証シミュレータ用Ｉ／Ｆ１７と、ハードウェア検証用シミュレータ用Ｉ／Ｆ１８と、システム検証シミュレータ用Ｉ／Ｆ１９と、デバッグ用Ｉ／Ｆ２０と、拡張用Ｉ／Ｆ２１とを備え、ブロックの機能モデル２２からなるシミュレータモデル２３として、検証用シミュレータ装置１を構成する。

【００２３】

検証用シミュレータ装置１がハードウェア検証用であれば、モデル９、モデル１６、モデル２３のそれぞれのハードウェア検証用シミュレータ用Ｉ／Ｆ４、ハードウェア検証用シミュレータ用Ｉ／Ｆ１１、ハードウェア検証用シミュレータ用Ｉ／Ｆ１８を用いてハードウェア検証用のシミュレータ制御手段２がシミュレーションの制御を行う。

【００２４】

また、検証用シミュレータ装置１がソフトウェア検証用であれば、モデル９、モデル１６、モデル２３のそれぞれのソフトウェア検証シミュレータ用Ｉ／Ｆ３、ソフトウェア検証シミュレータ用Ｉ／Ｆ１０、ソフトウェア検証シミュレータ用Ｉ／Ｆ１７を用いてソフトウェア検証用のシミュレータ制御手段２がシミュレーションの制御を行う。

【００２５】

さらに、検証用シミュレータ装置１がシステム検証用であれば、モデル９、モデル１６、モデル２３のそれぞれのシステム検証シミュレータ用Ｉ／Ｆ５、システム検証シミュレータ用Ｉ／Ｆ１２、システム検証シミュレータ用Ｉ／Ｆ１９を用いてシステム検証用のシミュレータ制御手段２がシミュレーションの制御を行

う。

【００２６】

それぞれの目的における検証時、内部の状態、例えばブロックのレジスタの値を知る必要がある。そのような場合には、デバッグ用Ｉ／Ｆ６、デバッグ用Ｉ／Ｆ１３、デバッグ用Ｉ／Ｆ２０を用いて、内部の状態を取り出し、デバッグに利用する。レジスタ以外の信号の値を含め、デバッグ目的で必要となる情報は、すべてこのＩ／Ｆにより実装される。

【００２７】

また、システムシミュレータにおいては、システムの振る舞いを基に、性能解析する必要が発生する。複数のバスマスタからなるマルチバスマスタシステムにおけるバスの負荷や、マスタがＣＰＵであった際のＣＰＵ利用率、メモリの転送レートや利用率が、そのような解析で必要となるデータである。各モデルは、バスの利用率を計算するために必要な、バス要求時やバス利用許可時、バス利用放棄時などを、このインターフェースを通して、シミュレータ制御手段をはじめとする解析系にデータを渡す。

【００２８】

簡単な解析系としては、インターフェースがそれらのデータを、電子データとしてファイルに保存し、それを利用して解析するものが挙げられる。

【００２９】

ブロック３１、ブロック３２、ブロック３３のレジスタの内容を記憶することをはじめとする機能は、機能モデル８、機能モデル１５、機能モデル２２でモデル化されるものとし、前述Ｉ／Ｆを用いて、機能を駆動する。レジスタへのアクセスや、デバッグのアクセスは、すべてこの機能モデルで実装される。このようなシミュレータ装置の実装をとることで、それぞれの目的ごとにモデルを用意する必要がなくなる。

【００３０】

図２は、被シミュレーション対象システムのＬＳＩであるシステムＬＳＩ３０の構成図である。システムＬＳＩ３０は、ブロック３１、ブロック３２、ブロック３３、ブロック３４、ブロック３５、ブロック３６、ブロック３７とバスから

構成される。

【００３１】

図３は、被シミュレーション対象システムのＬＳＩであるシステムＬＳＩ３０のハードウェア検証用シミュレータ６０の１実現例である。ハードウェア検証用シミュレータ６０は、システムＬＳＩ３０の構成要素ブロック３１、ブロック３２、ブロック３３、ブロック３４、ブロック３５、ブロック３６、ブロック３７とバスのそれぞれに対応して、検証用のモデル６１、モデル６２、モデル６３、モデル６４、モデル６５、モデル６６、モデル６７、およびバスモデル６８とバスモデル６９から構成されている。

【００３２】

検証対象ハードウェアがブロック３１であるとし、検証対象ハードウェアのモデルをモデル６１とする。ハードウェアのモデルは、ハードウェア記述言語（ＨＤＬ：Hardware Description Language）と呼ばれる記述言語、もしくは高位合成で設計されたモデルを含む。この目的のためには、検証対象ハードウェアブロックに影響を与える各モデルがクロックサイクルか、もしくはＲＴＬ（Register Transfer Level：レジスタ転送レベル）と呼ばれるレベルで検証対象ハードウェアブロックにアクセスする必要がある。

【００３３】

図６は、クロックレベルで必要とされる精度を示している。システムレベルの場合は、図中、時刻ｔ、時刻t_{+1}、時刻t_{+2}、時刻t_{+3}、時刻t_{+4}での各信号の値がピンレベルで、実際のハードウェア記述モデルと同じであると保証されていることが最低限必要となる。

【００３４】

図４（ａ）は、被シミュレーション対象システムのＬＳＩであるシステムＬＳＩ３０のソフトウェア検証用シミュレータ９０の１実現例である。ブロック３１が検証対象ソフトウェアが動作するブロックとし、そのモデルを９１とする。一般的には、ブロック３１のＩＳＳと呼ばれる命令レベルでの精度を保証する命令セットシミュレータと、ブロック３２、ブロック３３、ブロック３４、ブロック３５、ブロック３６、ブロック３７のメモリイメージをはじめとする、検証対象

ソフトウェアの動作に必要な機能を実装した仮想モデル９２から構成され、検証対象ソフトウェアがアクセスする部分は、仮想モデルとして実装される。ミドルウェアやデバイスドライバと呼ばれるハードウェアモデルの高精度を求めるソフトウェアが検証対象であると、この仮想モデルは、図３のハードウェア検証用シミュレータに近い精度が求められることもある。この目的で作られるシミュレータは、モデル９１がシミュレータの制御を行い、モデル９１の外部へアクセスが発生したときにのみ仮想モデル９２が使用される、といったモデル化がよく行われる。図４（ｂ）のような場合もある。

【００３５】

図７は、このレベルで必要とされる精度を示している。各命令が影響を与えるブロックに対する影響が、検証対象ソフトウェアの動作するブロック３１のモデル９１の内部ではなく、外部に、すなわち図の例では仮想モデル９２に及ぶことが最低限必要となる。

【００３６】

図５は、被シミュレーション対象システムのＬＳＩであるシステムＬＳＩ３０のシステム検証用シミュレータ１２０の１実現例である。ブロック３１、ブロック３２、ブロック３３、ブロック３４、ブロック３５、ブロック３６、ブロック３７のメモリイメージをはじめとする、検証対象システムの動作に必要な機能を実装したそれぞれのモデル、すなわち、モデル１２１、モデル１２２、モデル１２３、モデル１２４、モデル１２５、モデル１２６、モデル１２７とバスモデル１２８、バスモデル１２９から構成され、検証対象システム上で動作するソフトウェアがアクセスする部分は、全て実装される。

【００３７】

図８は、このレベルで必要とされる精度を示した一例である。図６と図７の間の精度を必要としている。図中にあるように、サイクルＣ、サイクルC_{+1}、サイクルC_{+2}、サイクルC_{+3}、サイクルC_{+4}の各サイクルで精度を保証する図のモデルは、その一例である。また、より抽象度の高いトランザクションでの精度を保証するシミュレータも、この範疇に含まれる。

【００３８】

ソフトウェア検証用シミュレータを、他の検証用シミュレータで利用するためには、まず、システムシミュレータ向けに、制御をシミュレータ制御手段２に行わせる、図９のようなソフトウェア検証シミュレータ用Ｉ／Ｆ３を用意すればよい。

【００３９】

従来のソフトウェア検証用シミュレータは、検証対象ソフトウェアが動作するプロセッサのみをモデル化し、バス、メモリに対するアクセスは抽象化されていた。一例を挙げると、図１０のフローチャートのような処理工程である。図１０中、初期化工程１００を行い、その後、ブロック３１のシミュレーション処理工程１０１を行う。このとき、バス、メモリに対する処理も、この処理工程内で行われる。そのため、図４（ｂ）に示すような複数のプロセッサからなるマルチプロセッサ構成では対応できない。

【００４０】

そこで、図９中、シミュレータ制御手段２により呼び出し可能な関数を定義し、その関数の実行時、外部へ影響を与える場合、前記ソフトウェア検証シミュレータ用Ｉ／Ｆ３を通して外部へアクセスを行う。

【００４１】

こうすることで、システムシミュレータとして利用することが可能となる。その処理工程を図１１のフローチャートに示す。図１１中、初期化工程１０５を行い、その後、シミュレーションが終わったかを工程１０６で調べる。シミュレーションが終わっていれば、シミュレーションを終了する。終わっていなければ、ブロック３１の機能モデル８の内部の値を更新するブロック３１のupdate処理工程１０７を行う。このとき、バス、メモリに影響を与える処理は行われない。次工程のブロック３１のcommunication処理工程１０８にて、機能モデル８外への処理を行う。このとき、関係するブロックのインターフェースを介してアクセスを行う。

【００４２】

以上、この処理工程を実装するものが、シミュレータ制御手段２である。上述ソフトウェア検証シミュレータ用Ｉ／Ｆ３の実装を行うことで、ソフトウェアシ

ミュレータは、シミュレータ装置の一部として利用可能になる。

【００４３】

このモデルをシステム検証用のシステムシミュレータで利用可能にするシステム検証シミュレータ用Ｉ／Ｆ５について説明する。前述したソフトウェア検証シミュレータ用Ｉ／Ｆ３は、機能モデル８を呼び出す。ソフトウェア検証用では、命令レベルの実装であったが、サイクルレベルで実装を行っている場合を考える。

【００４４】

命令レベルとサイクルレベルとの違いを図１２に示す。モデル３１が３つのパイプラインステージから構成されているとする。図にあるように、命令レベルでは、それぞれの命令が逐次的に実行される。

【００４５】

サイクルレベルでは、それぞれの命令がパイプラインステージを流れていく。サイクルＣでは命令１がステージ１で処理され、サイクルC_{+1}では、命令１がステージ２で、命令２がステージ１でそれぞれ処理されている。

【００４６】

図１３は、命令レベルでの処理工程を示すフローチャートである。命令レベルの処理工程のため、命令１の処理工程１０９、命令２の処理工程１１０、命令３の処理工程１１１の順に処理工程が進む。

【００４７】

機能モデル８では、図１３の命令１の処理工程１０９の処理は、図１４のフローチャートに図示する工程で処理を行う。すなわち、ステージ１での処理工程１０９０、ステージ２での処理工程１０９１、ステージ３での処理工程１０９２の順に、シミュレータ制御手段２が制御を行う。

【００４８】

同様にして、機能モデル８では、図１３の命令２の処理工程１１０の処理は、図１５のフローチャートに図示する工程で処理を行う。すなわち、ステージ１での処理工程１１００、ステージ２での処理工程１１０１、ステージ３での処理工程１１０２の順に、シミュレータ制御手段２が制御を行う。

【００４９】

また、同様にして、機能モデル８では、図１３の命令３の処理工程１１１の処理は、図１６のフローチャートに図示する工程で処理を行う。すなわち、ステージ１での処理工程１１１０、ステージ２での処理工程１１１１、ステージ３での処理工程１１１２の順に、シミュレータ制御手段２が制御を行う。

【００５０】

サイクルレベルの場合は、これら処理工程の制御方法を、図１７のフローチャートのように行えばよい。図１７は、前記機能モデル８について、図１２にあるサイクルＣ、サイクルC_{+1}、サイクルC_{+2}、サイクルC_{+3}、サイクルC_{+4}の処理に対応し、シミュレータ制御手段２が、システム検証用シミュレータの制御方法を行った処理工程を示すものである。

【００５１】

図中、Ｃ０がサイクルＣでの処理工程（命令１、ステージ１）を表し、Ｃ１がサイクルC_{+1}での処理（命令２、ステージ１と命令１、ステージ２）を表す。図中、Ｃ２ではサイクルC_{+2}の処理（命令３、ステージ１と命令２、ステージ２と命令１、ステージ３）を表し、Ｃ３は、サイクルC_{+3}における処理工程（命令３、ステージ２と命令２、ステージ３）を表し、Ｃ４は、サイクルC_{+4}における処理工程（命令３、ステージ３）を表している。

【００５２】

例えば処理工程Ｃ２にて、前記命令３のステージ１での処理工程１１１０、命令２のステージ２での処理工程１１０１、命令１のステージ３での処理工程１０９２を行っている一例である。ステージ間の依存関係によって、処理工程１１１０，１１０１，１０９２はどの順序で実行してもよく、逐次的にではなく、並列実行した場合にも、本発明の効果が得られることから、本発明の範疇に含まれる。

【００５３】

以上のように、シミュレータ制御手段２が、ブロック３１の機能モデル８との、システム検証シミュレータ用Ｉ／Ｆを用いた制御を行うことで、本発明で得られる効果が得られる。

【００５４】

次に、ブロック３１の機能モデル８をハードウェア検証用に用いるためのハードウェア検証用シミュレータ用Ｉ／Ｆ４について説明する。

【００５５】

ハードウェア検証用では、図６にある精度でのインターフェースが必要となる。最低限、クロックの立ち上がりと立ち下がりや、信号の値が変化したというイベントの受け渡しを、正しいタイミングで行うことを、このインターフェースでは実現する。上述したインターフェースの実装方法を用いることで、図８の精度を実現するシミュレータ制御手段２により、図８の精度を実現したシミュレータ装置が実現できる。

【００５６】

次に、この図８のレベルと、ハードウェア検証用のシミュレータが必要とするレベルの実装方法を説明する。

【００５７】

図１７での実装方法では、命令１のステージ１での処理工程１０９０は、図１１にあるupdateとcommunicateの実装を行っている。これら実装されたインターフェースは、シミュレータ制御手段２により、例えば、クロックサイクル毎に各モデルが呼び出され、機能モデル内部の状態を変更する関数updateと、外部への通信を行うフェーズであるcommunicateを実装することで、実現可能である。

【００５８】

ここで、ハードウェア検証用のシミュレータが必要とするのは、信号の駆動側がいつその信号をドライブするかといったタイミングである。システム検証用のシミュレータでは、信号の駆動側ではなく、トランザクションの駆動側（以降マスター）のタイミングで、communicateを実装している。

【００５９】

そこで、機能モデル８のインターフェースとして、トランザクションのマスターによるcommunicateを、図１８のフローチャートにおける信号を駆動するインターフェースを実装したcommunicateMaster処理工程１０８１と、駆動される信号に関するインターフェースを実装したcommunicateSlave処理工程１０８２に分

ける。

【００６０】

図１１の処理工程１０７と処理工程１０８が、図１２のサイクルC、サイクルC_{+1}、サイクルC_{+2}、サイクルC_{+3}、サイクルC_{+4}の各サイクル毎に呼び出される。そして、図１８では、処理工程１０８１、処理工程１０７、処理工程１０８２が、シミュレータ制御手段２により呼び出される。処理工程１０８のcommunicationでは、処理工程１０８１のcommunication masterと処理工程１０８２のcommunicationslaveを連続して呼ぶか、処理工程１０８１と処理工程１０８２での機能と同様の機能を、機能モデル８が提供すればよい。

【００６１】

以上、本発明の方法を用いることで、ソフトウェア検証向け、システム検証向け、ハードウェア検証向けでモデルの共通化が行える。本実施の形態では、ソフトウェア検証向け、システム検証向け、ハードウェア検証向けの抽象度として分かりやすい例を挙げたが、本発明はそれぞれの検証向けの、様々な抽象度を包含するものである。

【００６２】

次に、デバッグ用インターフェースに関して述べる。

【００６３】

モデルを用いたデバッグの例として、ＤＭＡブロックを対象とした場合について述べる。ＤＭＡはメモリのデータを転送する機能を実装したブロックである。すなわち、ブロックの機能としてデータ転送の開始、終了といった処理が行われた場合にブレークを行うようにするインターフェースを備えることで、デバッグ性が向上する。

【００６４】

このインターフェースは、機能に対してのものである場合、各用途向けモデルで共通して利用できる。さらに、デバッグでは、ブロックの持つメモリマップドＩＯレジスタを含むメモリの値を参照することがある。このメモリの値の参照、変更等を行うインターフェースも、このデバッグ用インターフェースに含まれる。各検証において、バスの振舞が必要でなく、メモリの値が書き換えられれば良

い場合があり、その場合には、このインターフェースを用いて高速にシミュレーションを実施することも可能となる。

【００６５】

ソフトウェア開発の場合など、更なる高速化のために、メモリ機能を、メモリを保持するブロックではなく、全てのメモリを管理するブロックに実装し、高速化を行う際には、このメモリブロックとソフトウェアが動作するブロックのみをシミュレータ制御手段から制御し、他のブロックをこの環境から分離することで、さらにシミュレーションを高速化することも可能であり、本発明の範疇に含まれる。

【００６６】

また、今までの例では、シミュレータ制御手段２が一つのレベルで実装する例を述べてきたが、実際の例では複数のレベルで実装される場合もある。図１９は、その一例である。シミュレータ制御手段２がシミュレータ制御手段Ｓ２０とシミュレータ制御手段Ｓ２１とシミュレータ制御手段Ｓ２２から構成される場合を示している。

【００６７】

シミュレータ制御手段Ｓ２０が、シミュレータ制御手段Ｓ２１とシミュレータ制御手段Ｓ２２を制御し、シミュレータ制御手段Ｓ２１がシミュレータモデル９とシミュレータモデル１６を制御する。このとき、シミュレーションモデル９とシミュレーションモデル１６を駆動するタイミングがクロックレベルである場合については、既に説明した。

【００６８】

シミュレータモデル９と１６で使用するクロックサイクルと、シミュレータモデル２３で使用するクロックサイクルが異なる場合、制御手段Ｓ２１とＳ２２では、それぞれのクロックサイクル数ではなく、シミュレーション時間で制御を行う必要があり、その機能を実装することで、異なるクロックソースを用いるモデル混在のシミュレーション環境が実現できる。

【００６９】

最後に拡張用インターフェースの例として、性能解析の例を挙げる。システム

検証では、どのバスマスタがどの程度バスを利用、占有しているか、いつどのメモリ番地にアクセスがあったか、どの程度バスの利用を待たされたかといった各種情報を利用して、性能解析を行う。このような情報を得るための拡張用インターフェースを用いて情報を得て性能解析を行い、性能解析の結果を基に再度システムをチューニングして、最適なシステムを設計することが可能となる。

【００７０】

【発明の効果】

以上のように本発明によれば、大規模なシステムＬＳＩのハードウェア検証用システムシミュレータ、ソフトウェア開発検証用シミュレータ、システム検証用システムシミュレータをはじめとする各種目的用システムシミュレータにおいて、共通のモデルが利用可能になるため、異なるシミュレータ間でモデルの共有化が図られ、統合シミュレーション制御方法によって、適切な精度のシミュレーションを同一のモデルを用いて実行できる。これによって、大規模なシステムＬＳＩのハードウェア／ソフトウェア協調設計環境を構築するための工数を削減でき、ひいてはシステムＬＳＩの設計期間短縮につながり、高性能、低コストのシステムＬＳＩを提供することができる。

【図面の簡単な説明】

【図１】　本発明の実施の形態のシミュレータ装置の構成を示すブロック図

【図２】　システムＬＳＩを示す概念図

【図３】　ハードウェア検証用シミュレータを示す概念図

【図４】　ソフトウェア検証用シミュレータを示す概念図

【図５】　システム検証用シミュレータを示す概念図

【図６】　ハードウェア検証用シミュレータで必要とされる精度を示す概念図

【図７】　ソフトウェア検証用シミュレータで必要とされる精度を示す概念図

【図８】　システム検証用シミュレータで必要とされる精度を示す概念図

【図９】　ソフトウェア検証用シミュレータＩ／Ｆを示す概念図

【図１０】　従来のソフトウェア検証用シミュレータの処理工程を示すフローチャート

【図１１】　システムシミュレータの処理工程を示すフローチャート

【図１２】　命令レベルとサイクルレベルとの違いの一部を示す説明図

【図１３】　命令レベルでの処理工程を示すフローチャート

【図１４】　命令レベルでの命令１の処理工程を示すフローチャート

【図１５】　命令レベルでの命令２の処理工程を示すフローチャート

【図１６】　命令レベルでの命令３の処理工程を示すフローチャート

【図１７】　サイクルレベルでの処理工程を示すフローチャート

【図１８】　ハードウェア検証用シミュレータの処理工程を示すフローチャート

【図１９】　本発明の別の実施の形態のシミュレータ装置の構成を示すブロック図

【符号の説明】

１　検証用シミュレータ装置

２　シミュレータ制御手段

３，１０，１７　ソフトウェア検証シミュレータ用Ｉ／Ｆ

４，１１，１８　ハードウェア検証用シミュレータ用Ｉ／Ｆ

５，１２，１９　システム検証シミュレータ用Ｉ／Ｆ

６，１３，２０　デバッグ用Ｉ／Ｆ

７，１４，２１　拡張用Ｉ／Ｆ

８，１５，２２　ブロックの機能モデル

９，１６，２３　シミュレータモデル

３１，３２，３３，３４，３５，３６，３７　ブロック

６０　ハードウェア検証用シミュレータ

６１，６２，６３，６４，６５，６６，６７　ハードウェア対応の検証用モデル

６８，６９　バスモデル

９０，９５　ソフトウェア検証用シミュレータ

９１，９２　ソフトウェア対応の検証用モデル

９２，９４　仮想モデル

１２０　システム検証用シミュレータ

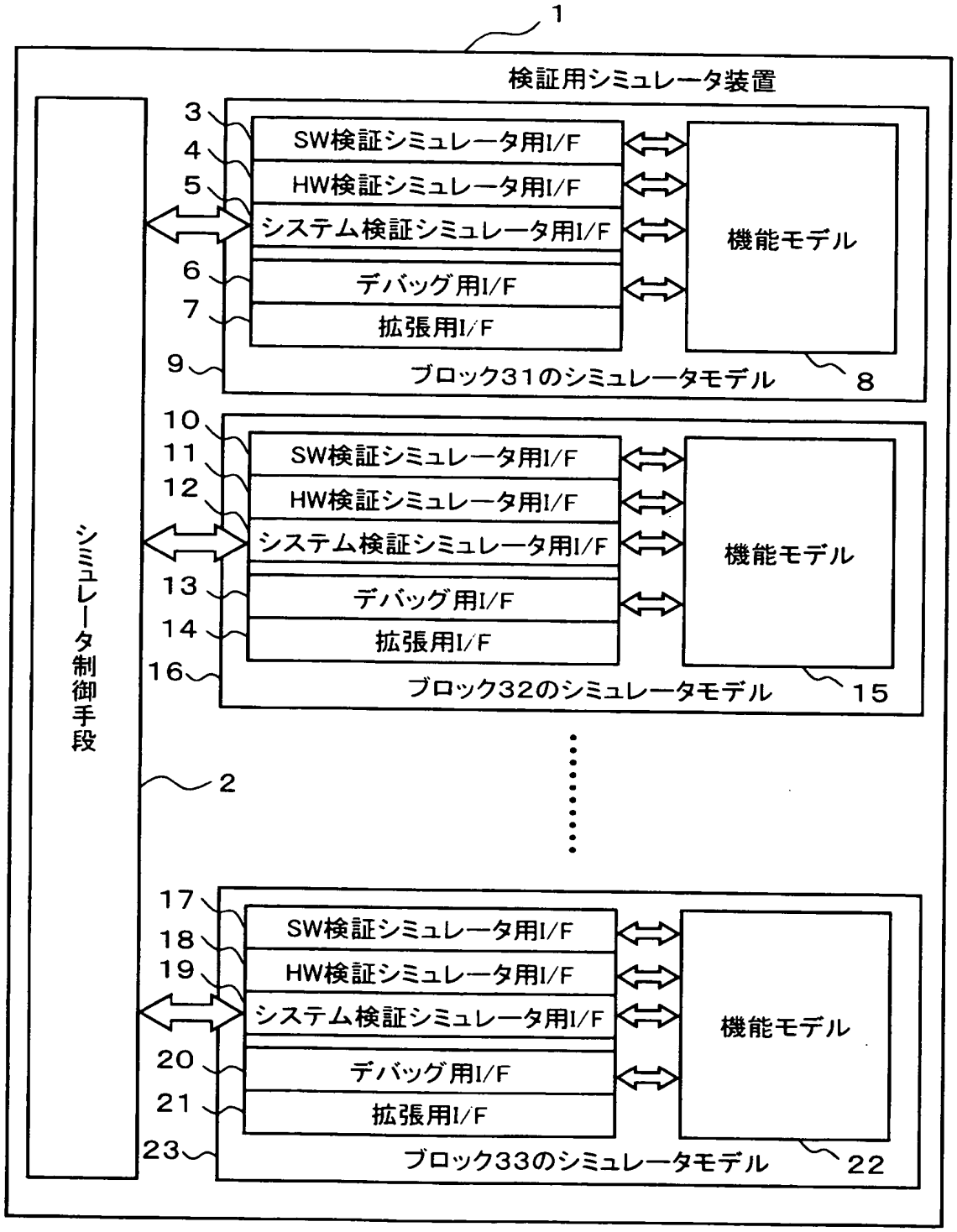
１２１，１２２，１２３，１２４，１２５，１２６，１２７　システム対応の検証用モデル

１２８，１２９　バスモデル

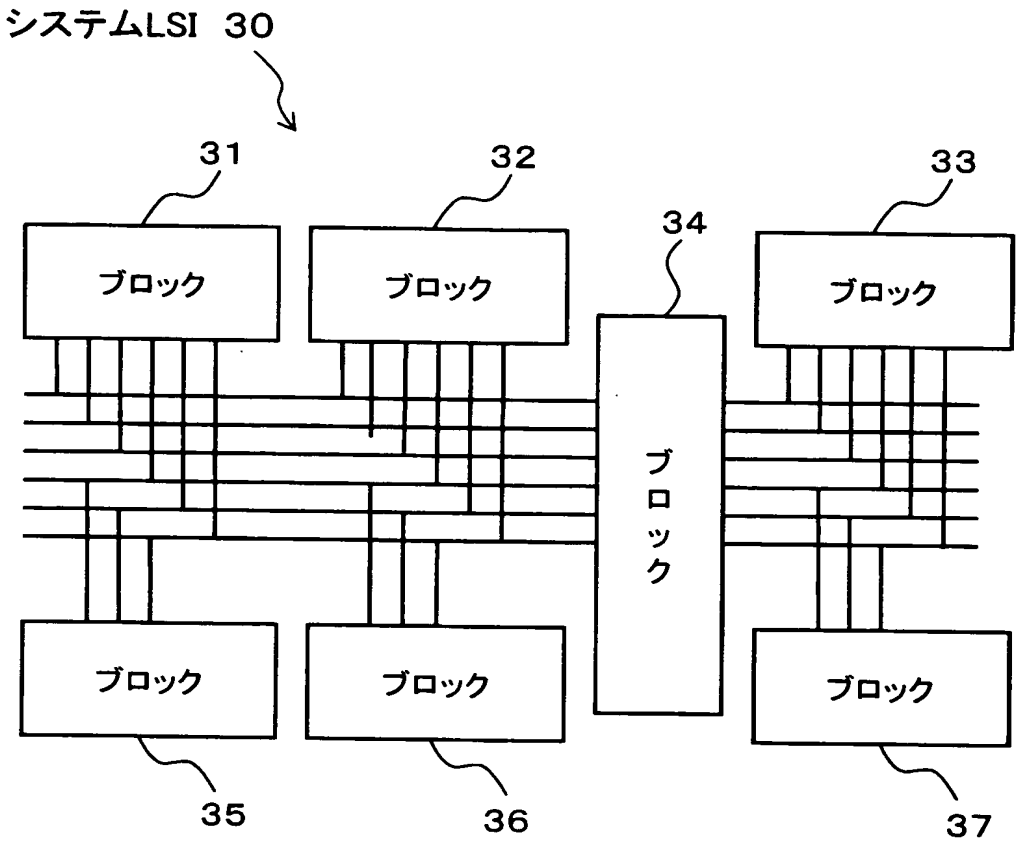
Ｓ２１，Ｓ２２，Ｓ２３　シミュレータ制御手段

【書類名】　図面

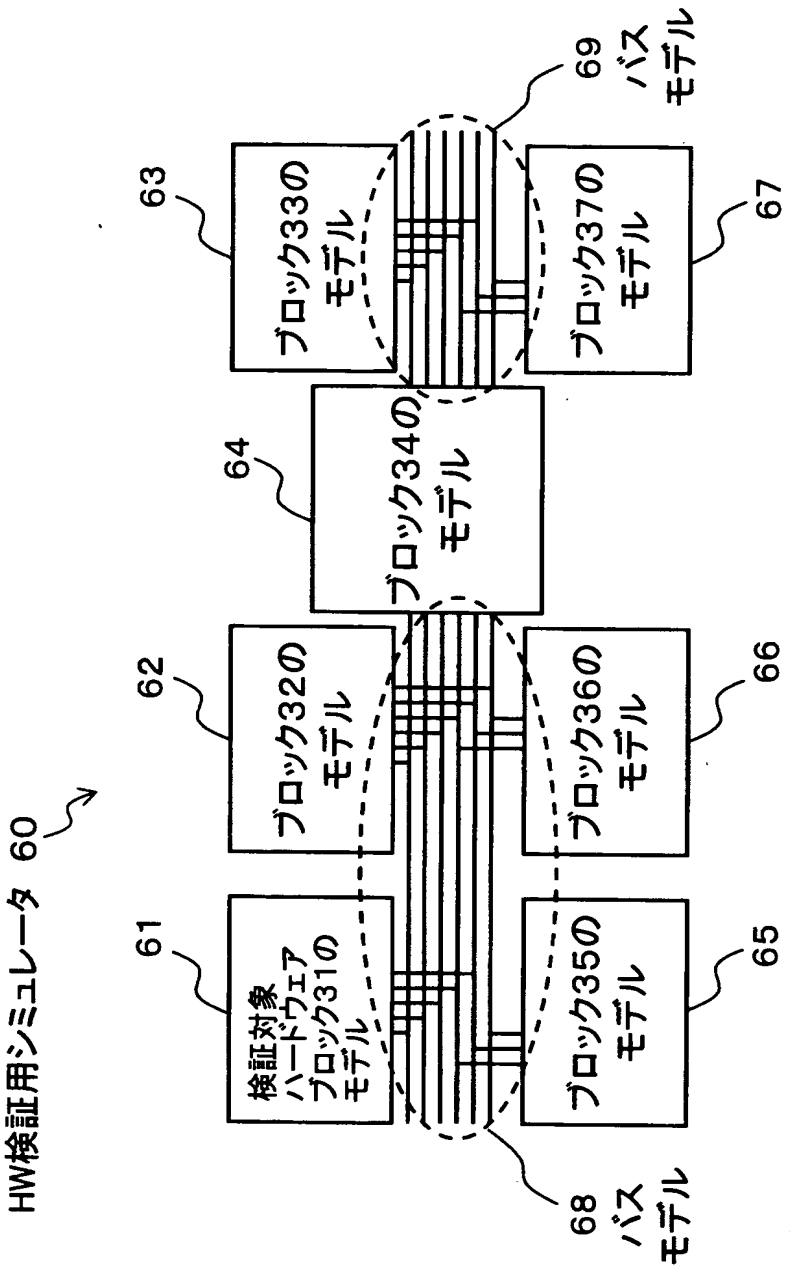
【図１】


1
検証用シミュレータ装置
シミュレータ制御手段
2
3
4
5
6
7
9
SW検証シミュレータ用I/F
HW検証シミュレータ用I/F
システム検証シミュレータ用I/F
デバッグ用I/F
拡張用I/F
機能モデル
8
ブロック31のシミュレータモデル
10
11
12
13
14
16
15
ブロック32のシミュレータモデル
17
18
19
20
21
23
22
ブロック33のシミュレータモデル

【図２】

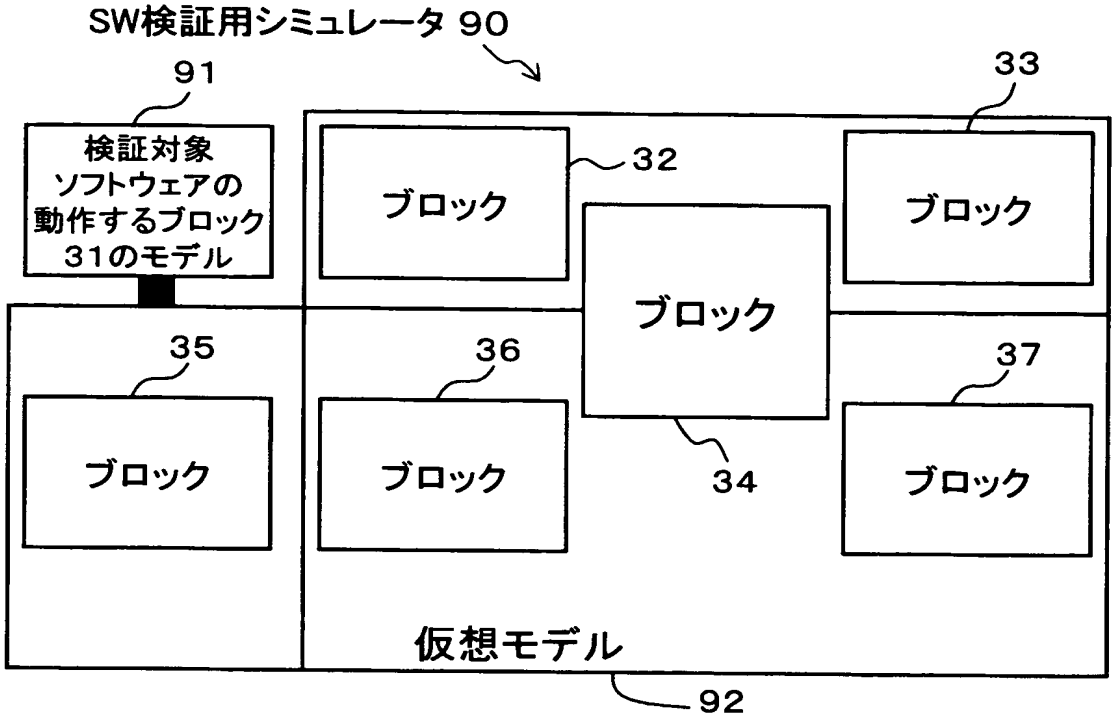

システムLSI 30
31
32
33
34
ブロック
ブロック
ブロック
ブロック
ブロック
ブロック
ブロック
35
36
37

【図３】

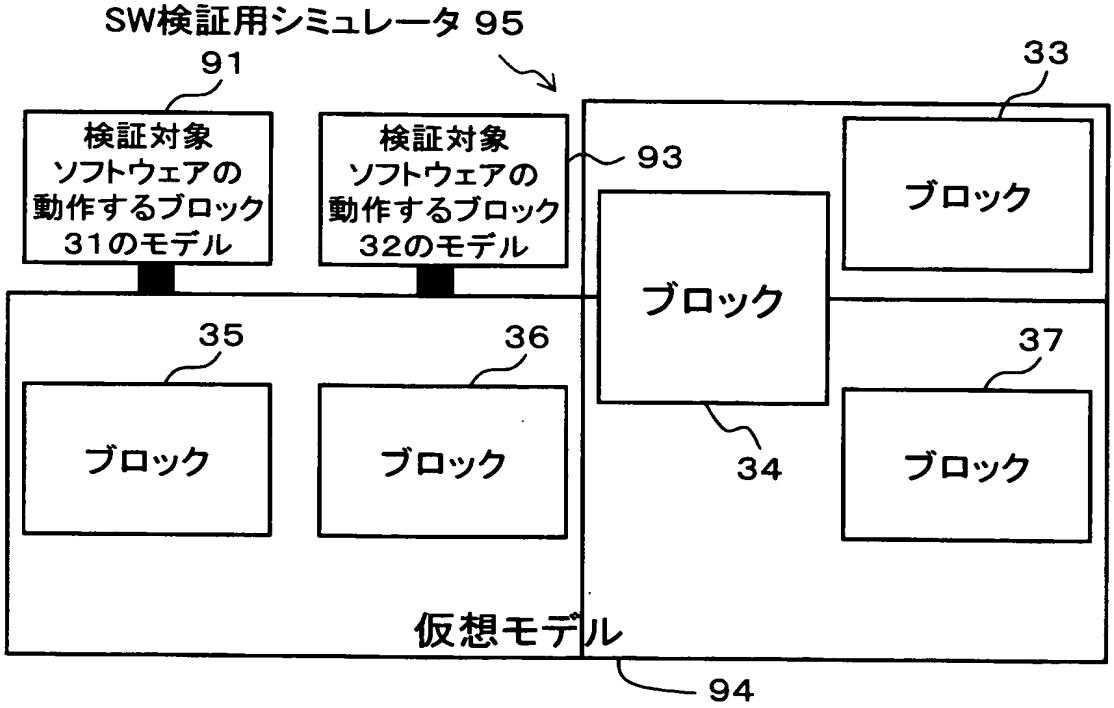

HW検証用シミュレータ 60
検証対象ハードウェアブロック31のモデル 61
ブロック32のモデル 62
ブロック33のモデル 63
ブロック34のモデル 64
ブロック35のモデル 65
ブロック36のモデル 66
ブロック37のモデル 67
バスモデル 68
バスモデル 69

【図４】

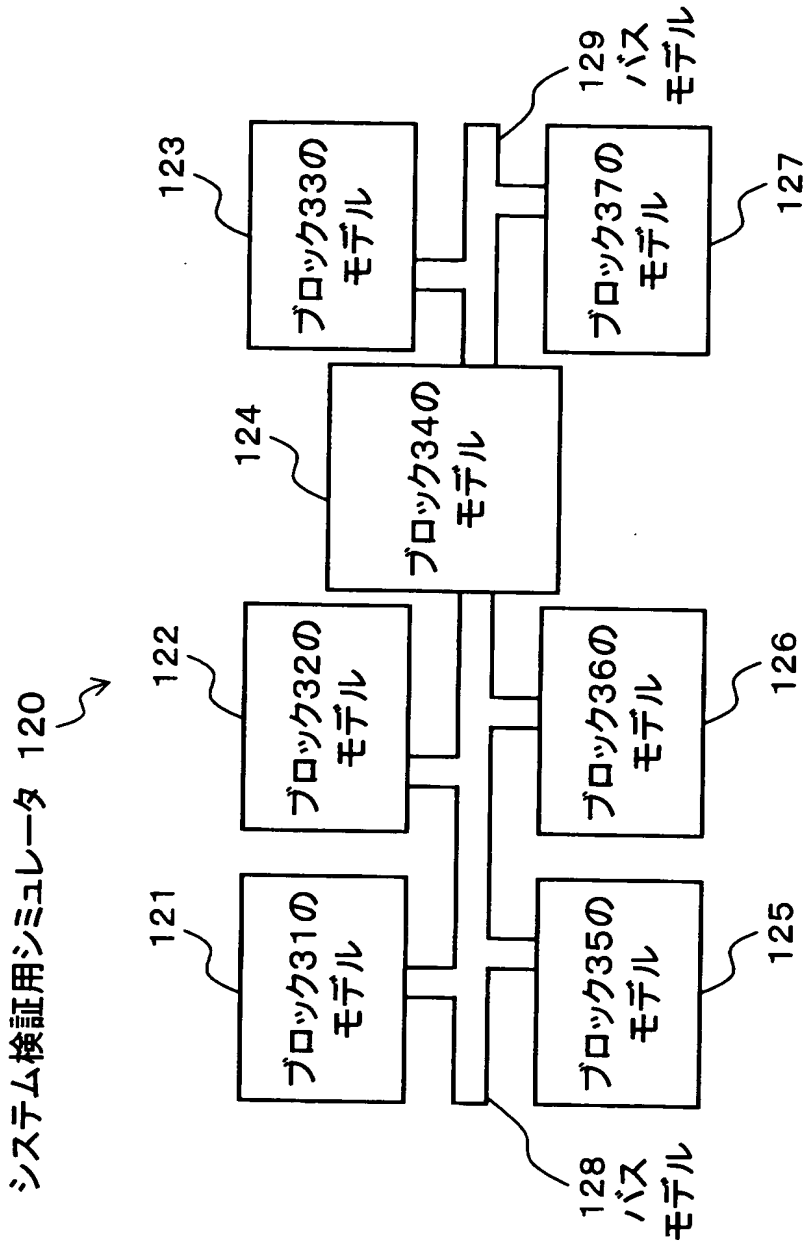
(a)


SW検証用シミュレータ 90
91
検証対象
ソフトウェアの
動作するブロック
31のモデル
32
ブロック
33
ブロック
ブロック
35
ブロック
36
ブロック
34
37
ブロック
仮想モデル
92

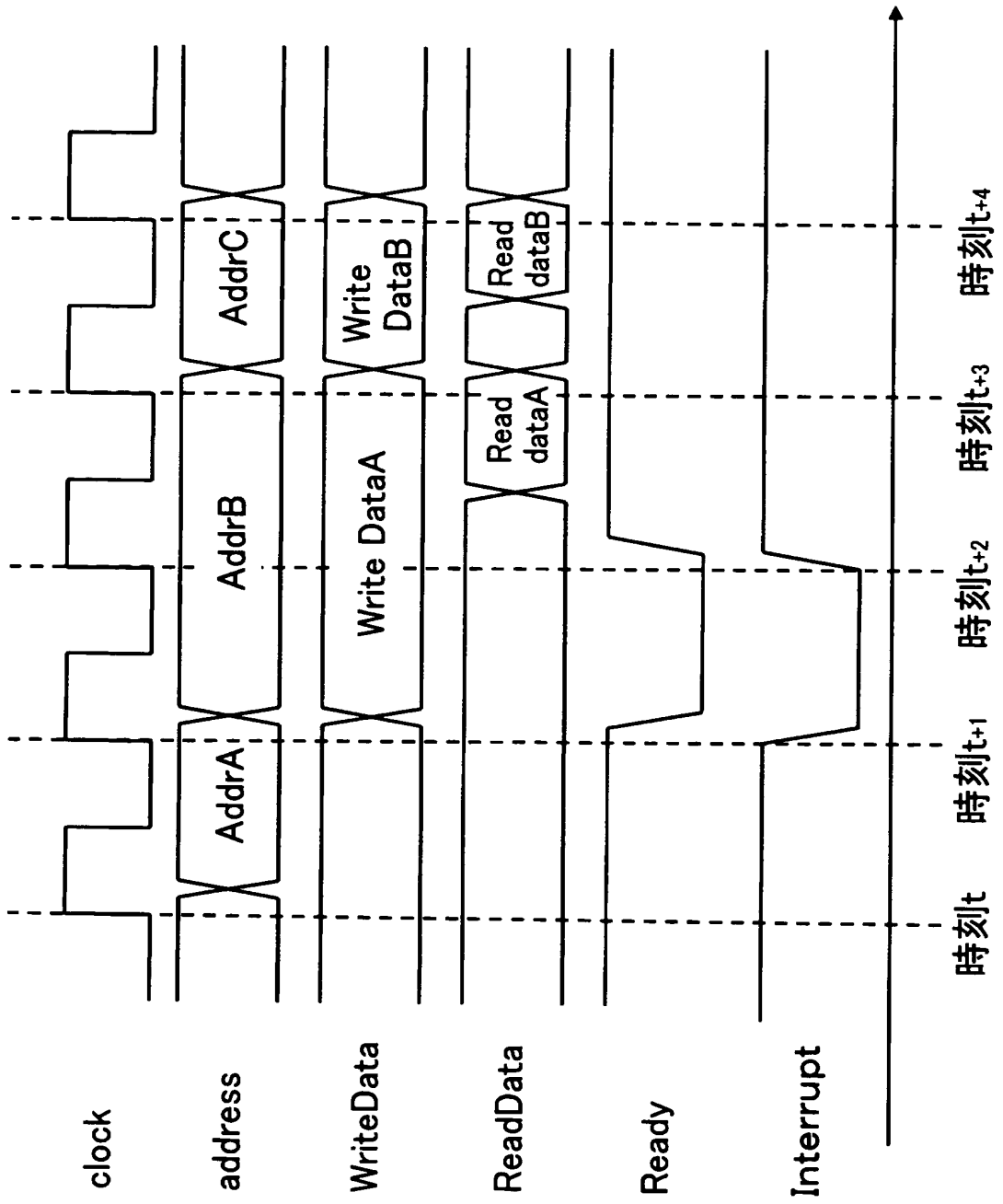
(b)


SW検証用シミュレータ 95
91
検証対象
ソフトウェアの
動作するブロック
31のモデル
検証対象
ソフトウェアの
動作するブロック
32のモデル
93
33
ブロック
ブロック
35
ブロック
36
ブロック
34
37
ブロック
仮想モデル
94

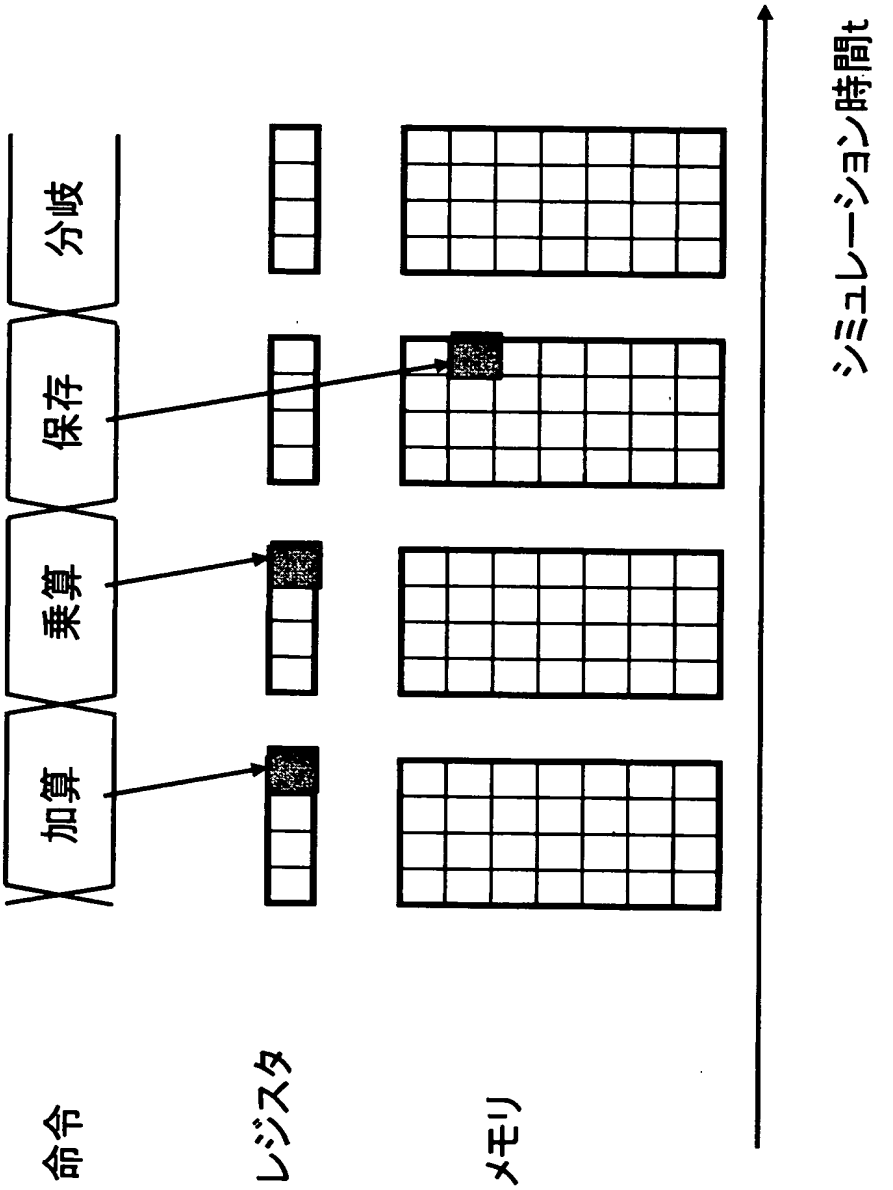
【図５】


システム検証用シミュレータ 120
ブロック31のモデル
121
ブロック32のモデル
122
ブロック33のモデル
123
ブロック34のモデル
124
ブロック35のモデル
125
ブロック36のモデル
126
ブロック37のモデル
127
128 バスモデル
129 バスモデル

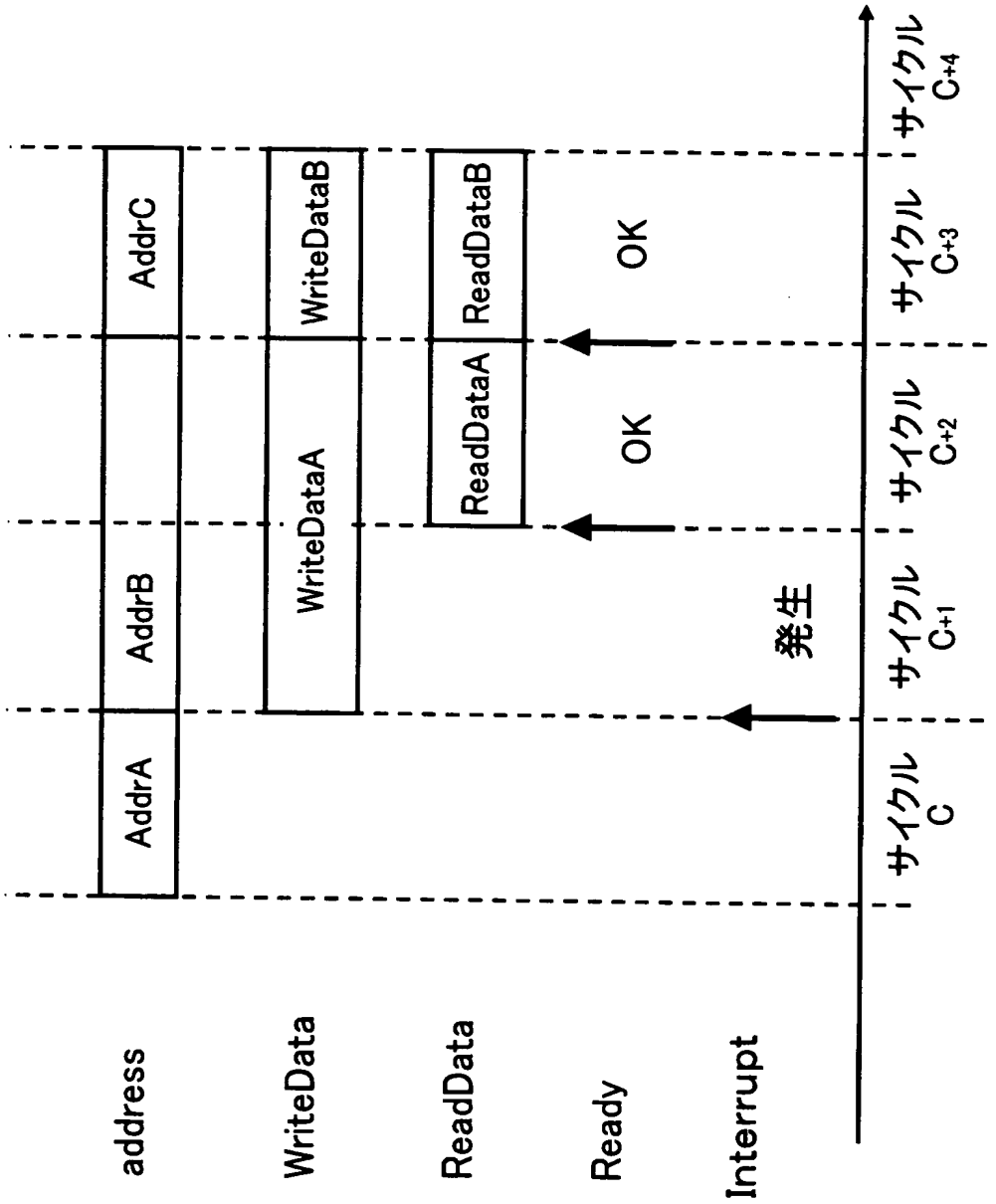
【図６】


clock
address
WriteData
ReadData
Ready
Interrupt
AddrA
AddrB
AddrC
Write DataA
Write DataB
Read dataA
Read dataB
時刻t
時刻t+1
時刻t+2
時刻t+3
時刻t+4

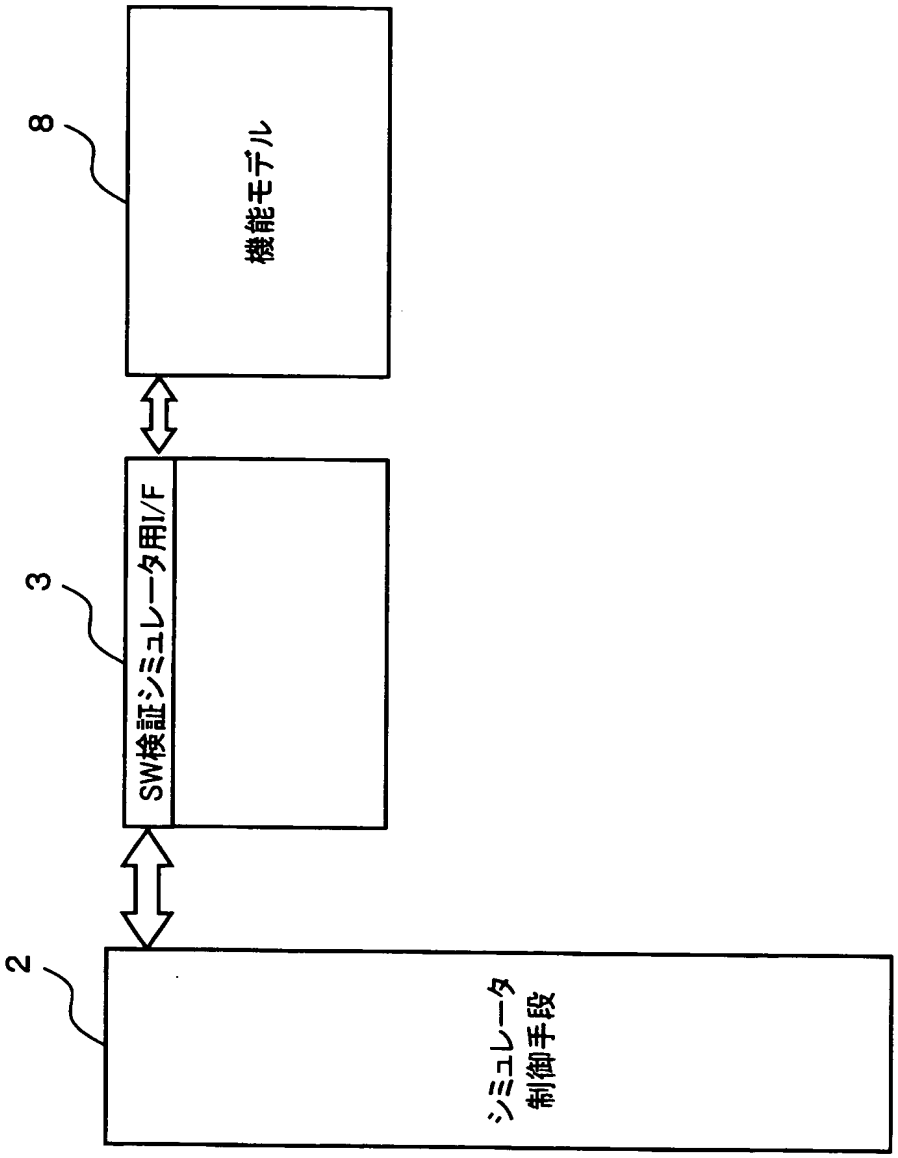
【図７】



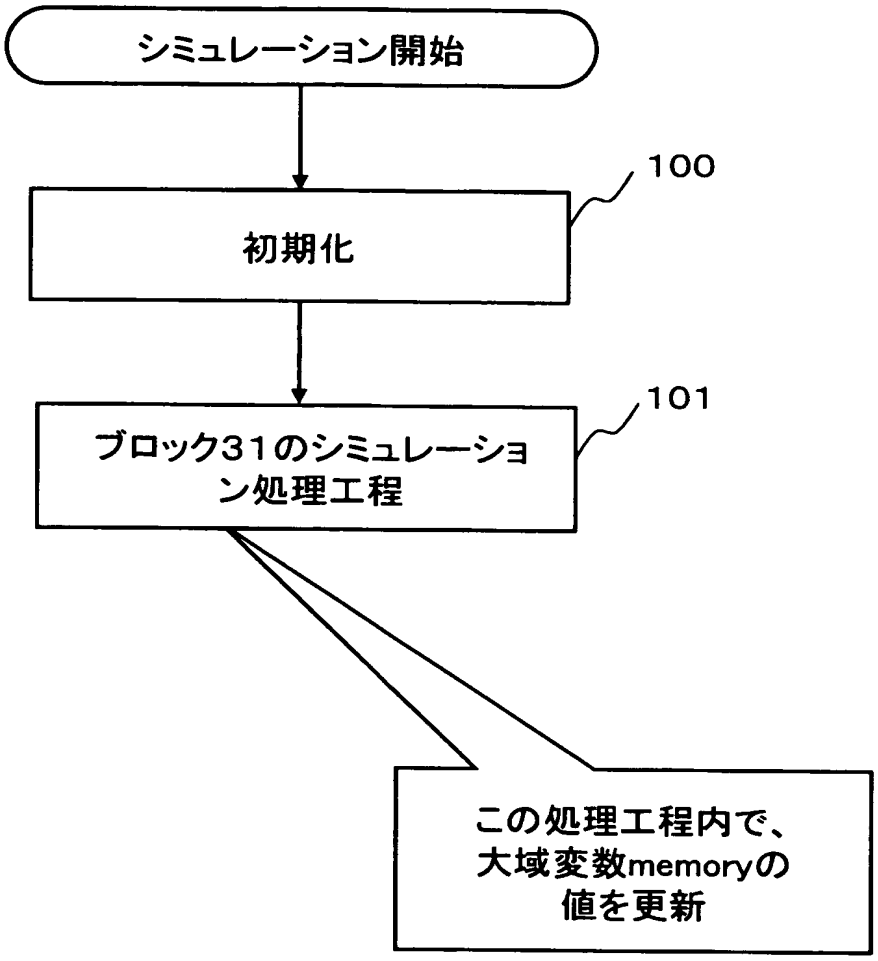
【図８】


address
WriteData
ReadData
Ready
Interrupt
AddrA
AddrB
AddrC
WriteDataA
WriteDataB
ReadDataA
ReadDataB
OK
OK
発生
サイクル C
サイクル C+1
サイクル C+2
サイクル C+3
サイクル C+4

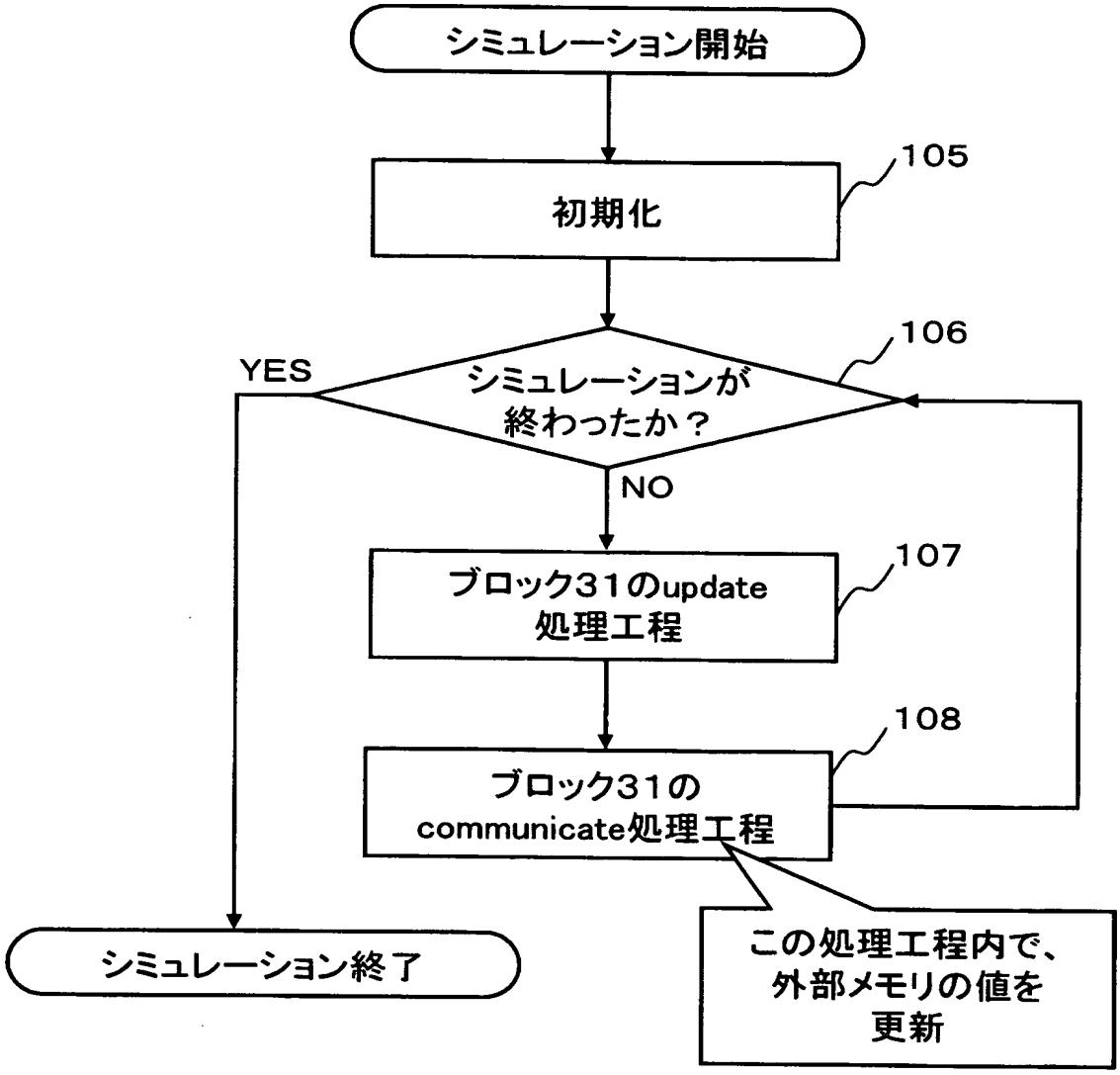
【図９】



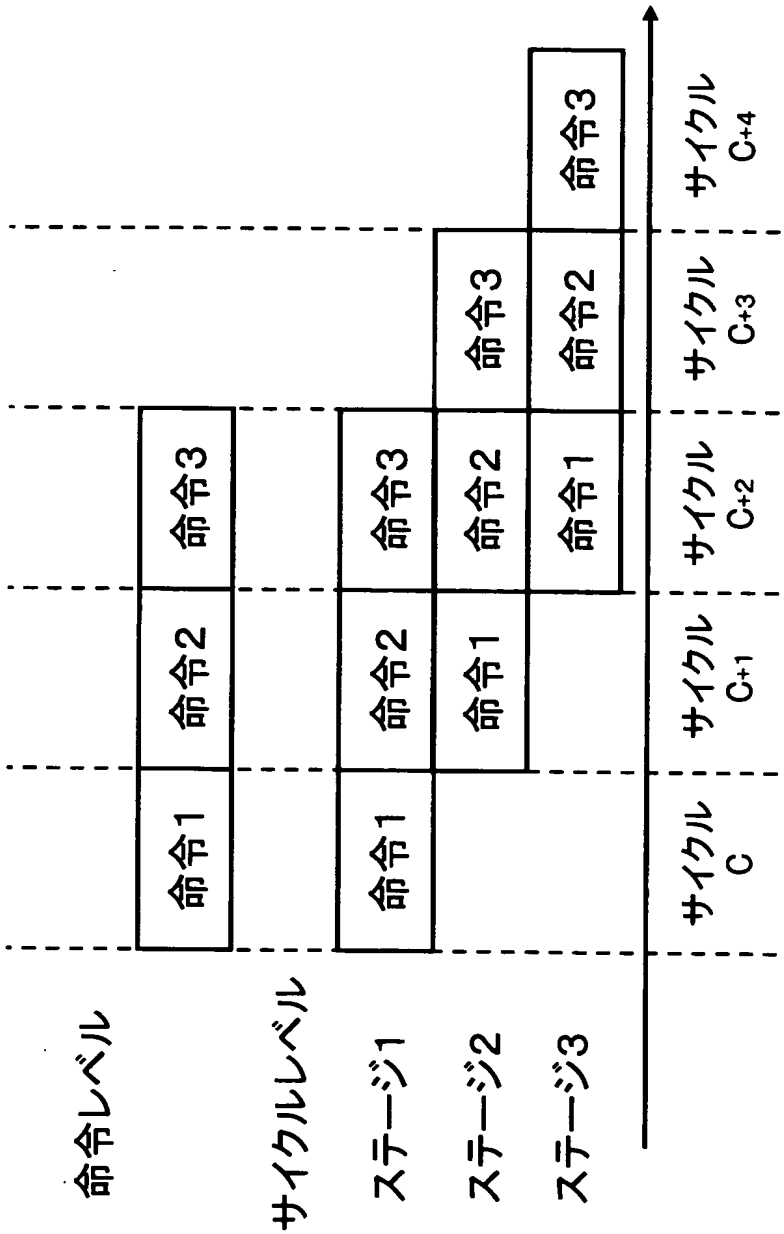
【図１０】


シミュレーション開始
100
初期化
101
ブロック31のシミュレーション処理工程
この処理工程内で、大域変数memoryの値を更新

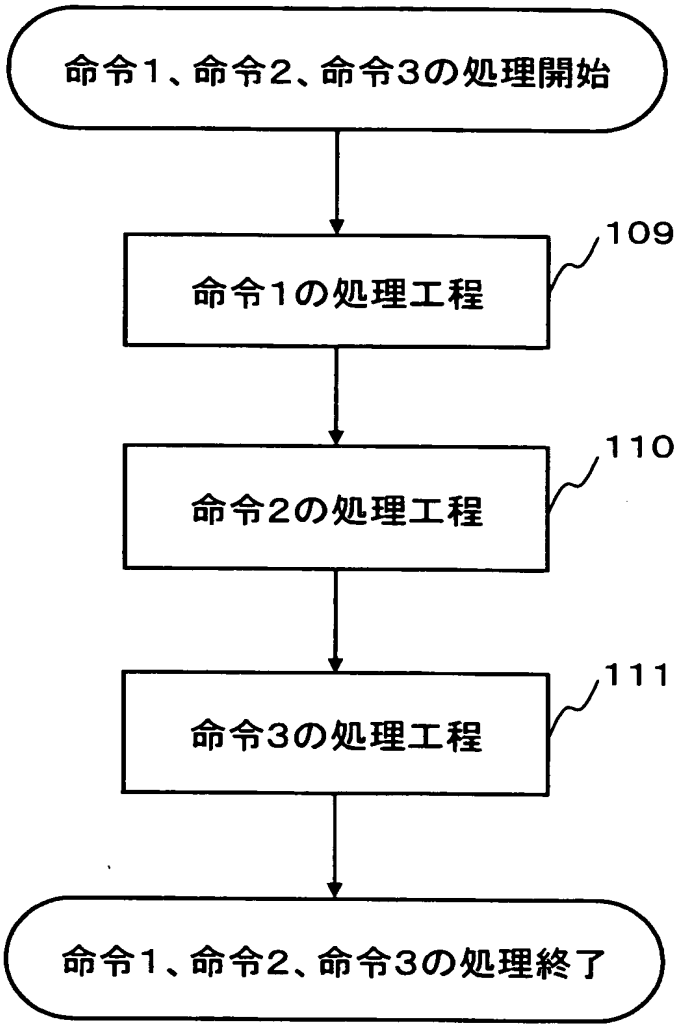
【図１１】


シミュレーション開始
105
初期化
106
YES
シミュレーションが
終わったか？
NO
107
ブロック31のupdate
処理工程
108
ブロック31の
communicate処理工程
この処理工程内で、
外部メモリの値を
更新
シミュレーション終了

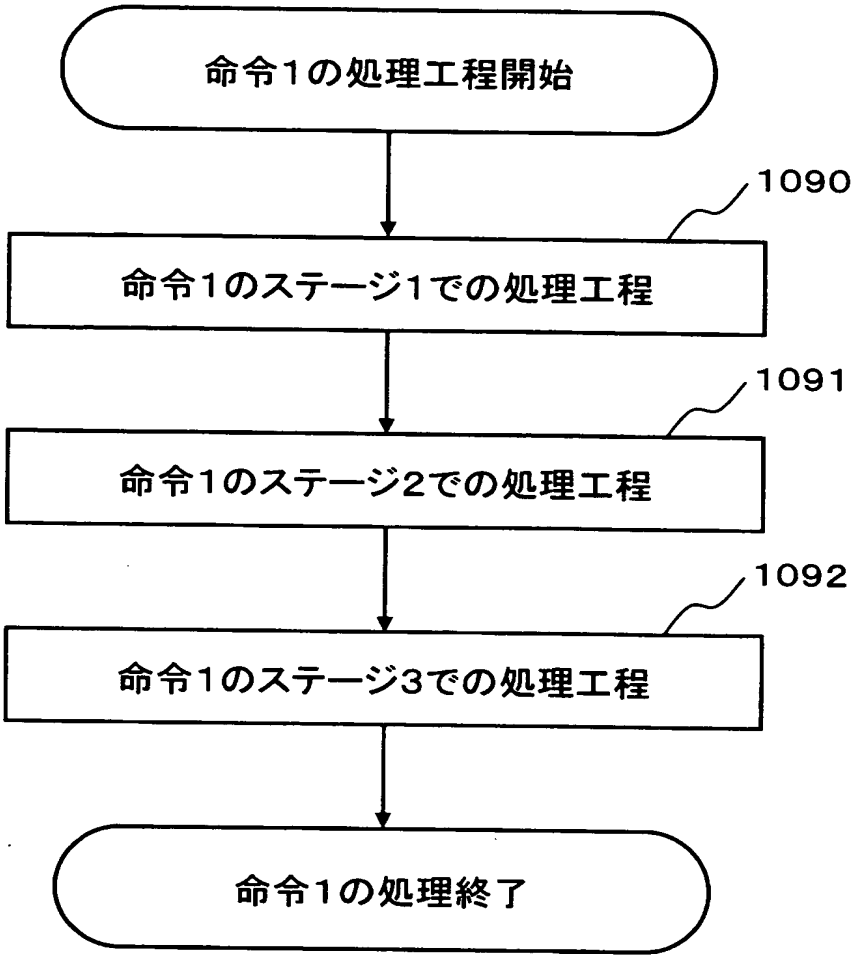
【図１２】


命令レベル
サイクルレベル
ステージ1
ステージ2
ステージ3
命令1
命令2
命令3
サイクル C
サイクル C+1
サイクル C+2
サイクル C+3
サイクル C+4

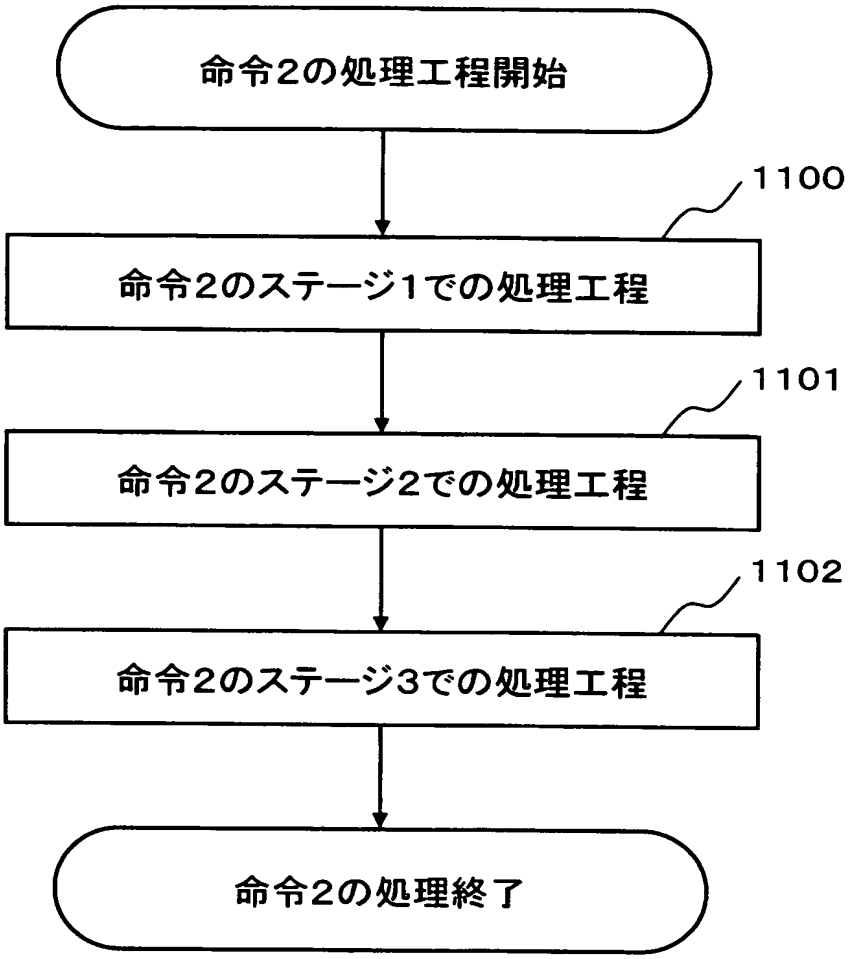
【図１３】


命令1、命令2、命令3の処理開始
命令1の処理工程
109
命令2の処理工程
110
命令3の処理工程
111
命令1、命令2、命令3の処理終了

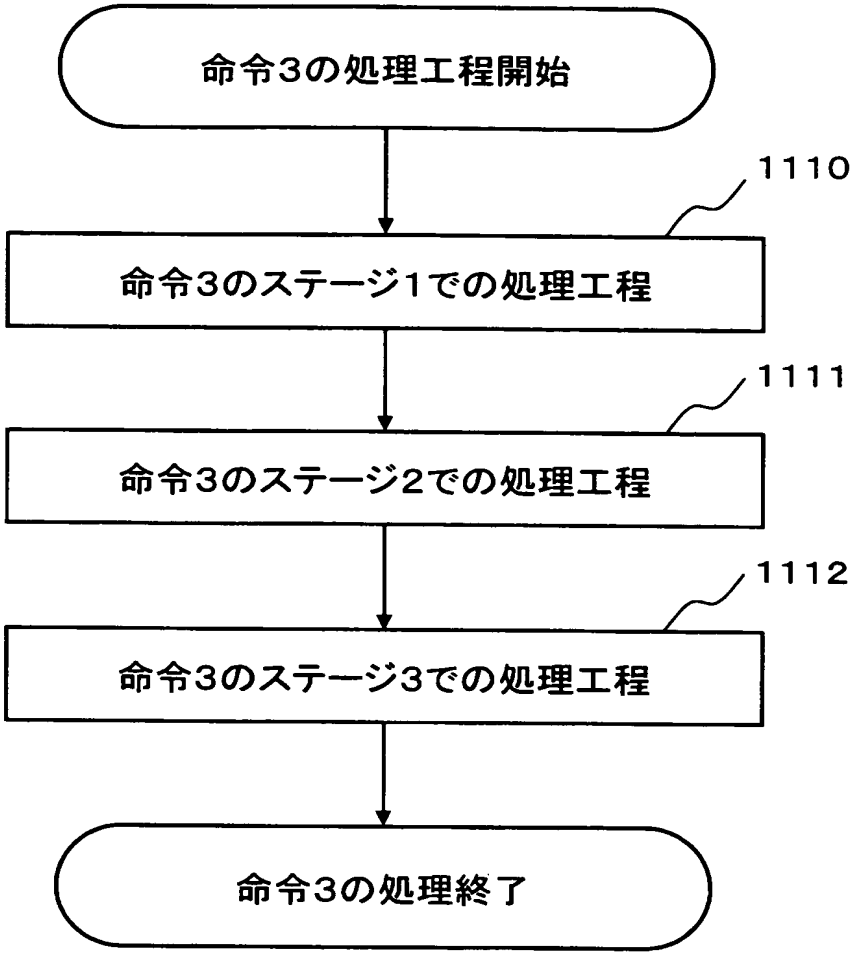
【図１４】


命令1の処理工程開始
1090
命令1のステージ1での処理工程
1091
命令1のステージ2での処理工程
1092
命令1のステージ3での処理工程
命令1の処理終了

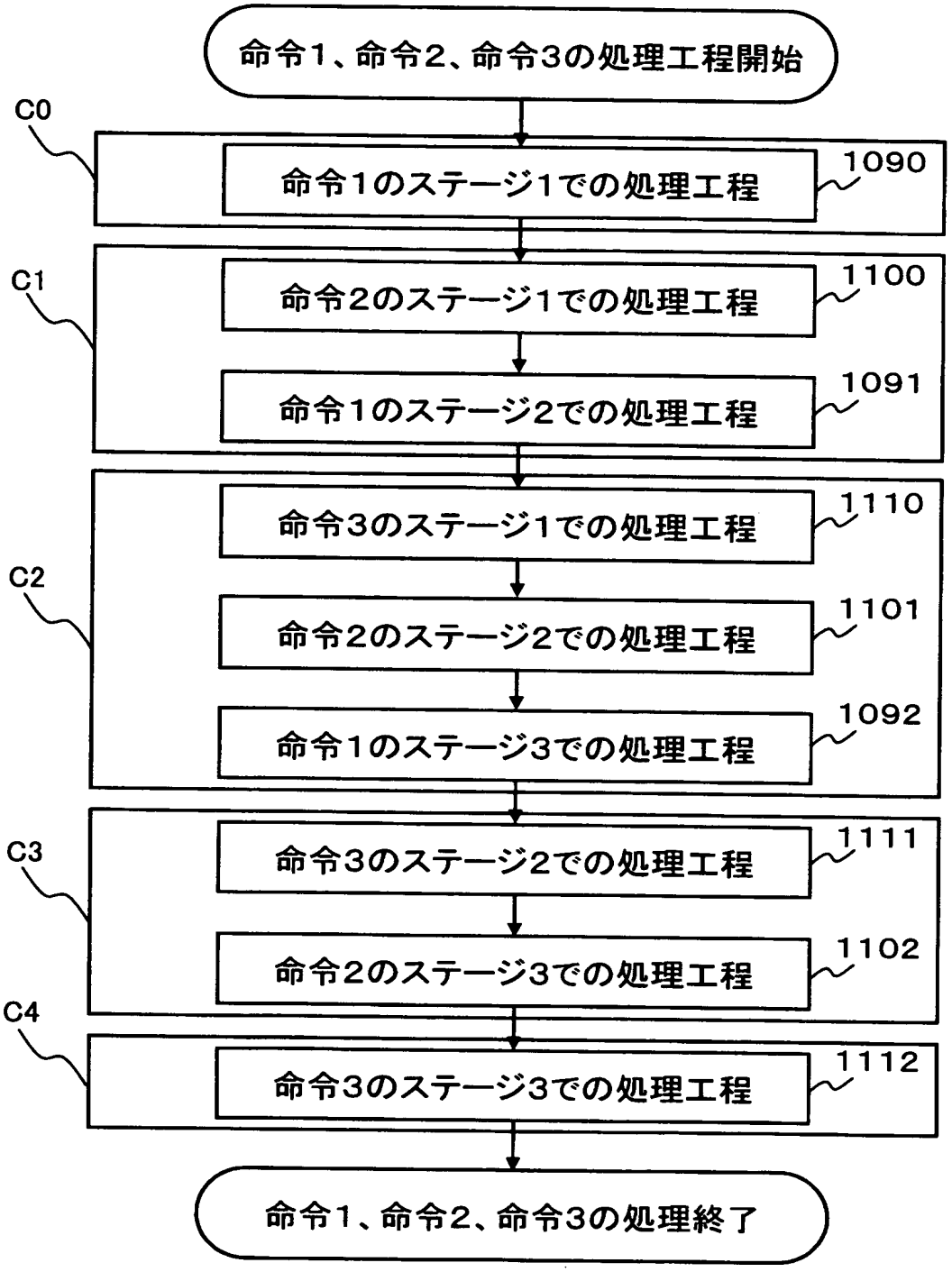
【図１５】


命令2の処理工程開始
1100
命令2のステージ1での処理工程
1101
命令2のステージ2での処理工程
1102
命令2のステージ3での処理工程
命令2の処理終了

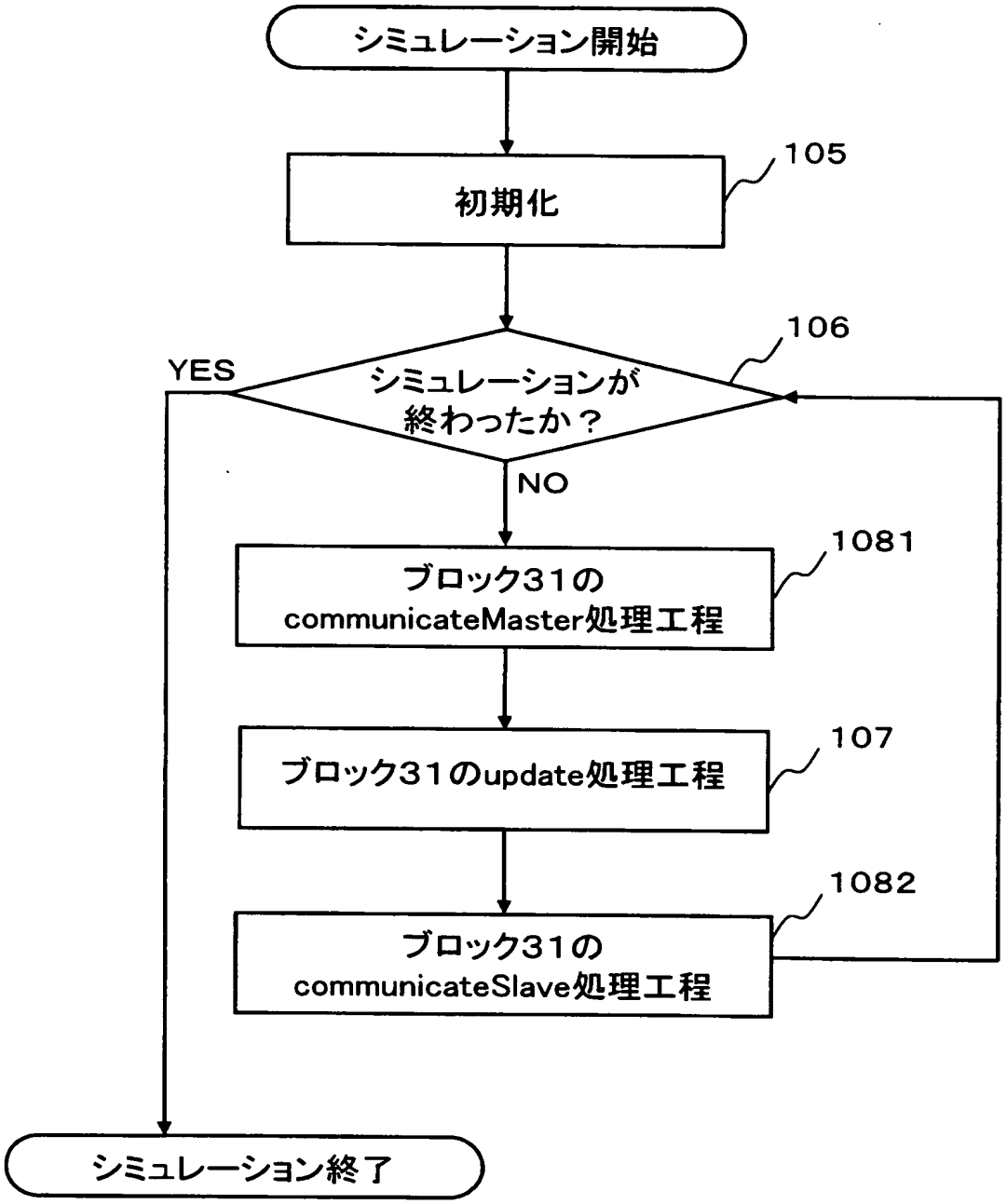
【図１６】


命令3の処理工程開始
1110
命令3のステージ1での処理工程
1111
命令3のステージ2での処理工程
1112
命令3のステージ3での処理工程
命令3の処理終了

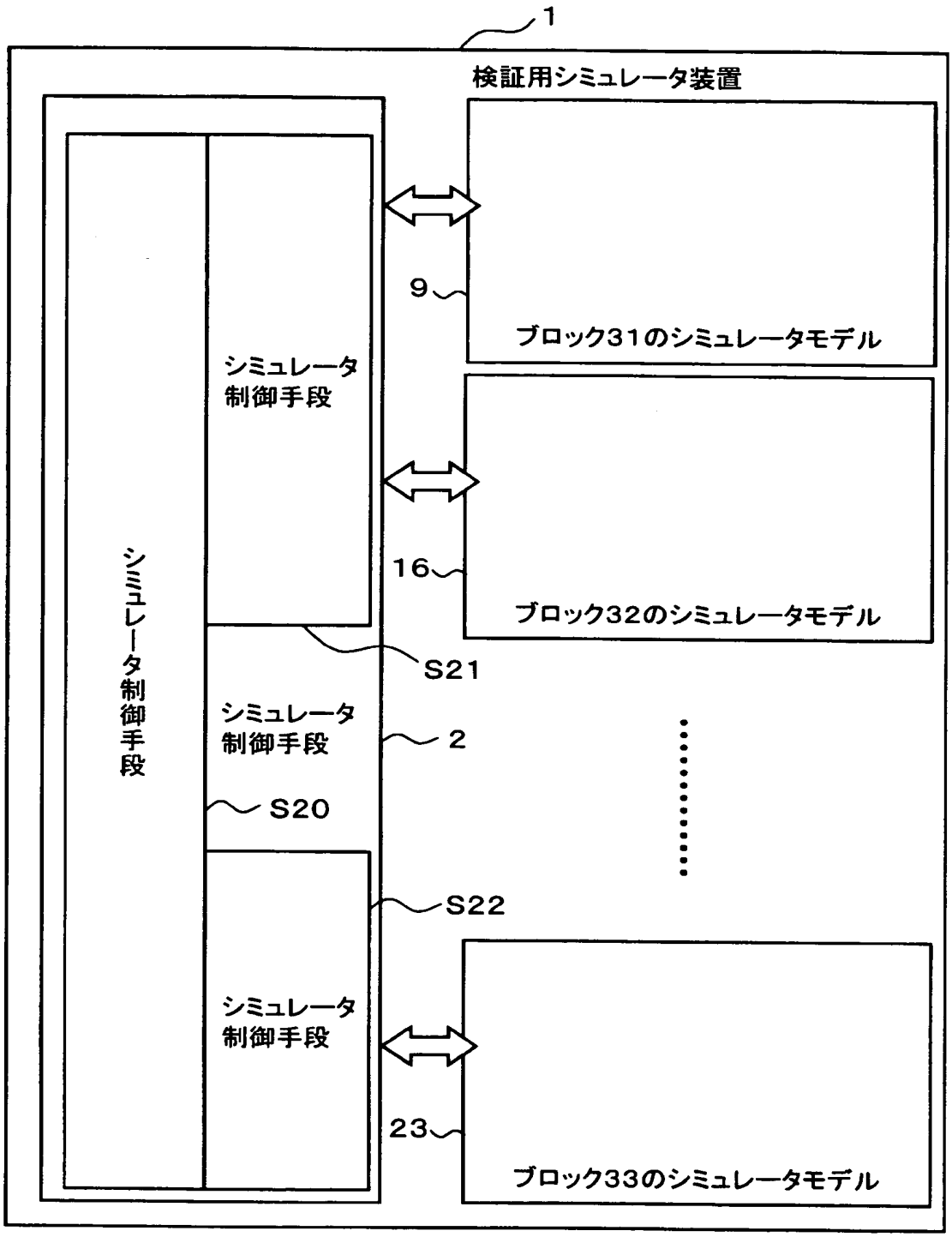
【図１７】


命令1、命令2、命令3の処理工程開始
C0
命令1のステージ1での処理工程
1090
C1
命令2のステージ1での処理工程
1100
命令1のステージ2での処理工程
1091
C2
命令3のステージ1での処理工程
1110
命令2のステージ2での処理工程
1101
命令1のステージ3での処理工程
1092
C3
命令3のステージ2での処理工程
1111
命令2のステージ3での処理工程
1102
C4
命令3のステージ3での処理工程
1112
命令1、命令2、命令3の処理終了

【図１８】


シミュレーション開始
105
初期化
106
YES
シミュレーションが
終わったか？
NO
1081
ブロック31の
communicateMaster処理工程
107
ブロック31のupdate処理工程
1082
ブロック31の
communicateSlave処理工程
シミュレーション終了

【図１９】


1
検証用シミュレータ装置
シミュレータ制御手段
シミュレータ制御手段
9
ブロック31のシミュレータモデル
16
ブロック32のシミュレータモデル
S21
シミュレータ制御手段
2
S20
S22
シミュレータ制御手段
23
ブロック33のシミュレータモデル

【書類名】　要約書

【要約】

【課題】　システムＬＳＩ開発の、それぞれの用途向けにシステムＬＳＩを構成するブロックのモデルを用意するのが現状であるが、システムＬＳＩの複雑化大規模化を鑑みると、それらのモデルを用意するための工数も同様に増加している。結果、ハードウェア／ソフトウェア協調設計の利点が失われてしまう。

【解決手段】　大規模なシステムＬＳＩの１チップシミュレーションで用いるブロックを、異なるシミュレータの制御部から呼び出し可能なインターフェースを実装することで、シミュレータ設計の工数を削減する。また、デバッグ用のインターフェースを実装することで、各用途のための検証解析機能を実現する。

【選択図】　図１

認定・付加情報

特許出願の番号	特願２００３－１０４９０７
受付番号	５０３００５８５４４７
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１５年　４月１０日

＜認定情報・付加情報＞

【提出日】　平成15年 4月 9日

次頁無

特願２００３－１０４９０７

出　願　人　履　歴　情　報

識別番号　　[０００００５８２１]

１．変更年月日　　１９９０年　８月２８日
[変更理由]　　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社